US009197486B2

(12) **United States Patent**
Roskind

(10) **Patent No.:** **US 9,197,486 B2**
(45) **Date of Patent:** ***Nov. 24, 2015**

(54) ADAPTIVE ACCELERATED APPLICATION STARTUP

(75) Inventor: **James Roskind**, Redwood City, CA (US)

(73) Assignee: **Google Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 781 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **12/201,979**

(22) Filed: **Aug. 29, 2008**

(65) **Prior Publication Data**

US 2010/0057936 A1 Mar. 4, 2010

(51) **Int. Cl.**
H04L 29/12 (2006.01)
H04L 29/08 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 29/12066*** (2013.01); ***H04L 61/1511*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/28*** (2013.01); ***H04L 67/289*** (2013.01); ***H04L 67/2842*** (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,649,196	A *	7/1997	Woodhill et al.	711/148
5,774,660	A *	6/1998	Brendel et al.	709/201
5,923,848	A	7/1999	Goodhand et al.	
6,016,512	A *	1/2000	Huitema	709/245
6,023,726	A	2/2000	Saksena	
6,118,784	A *	9/2000	Tsuchiya et al.	370/401
6,393,605	B1 *	5/2002	Loomans	717/121
6,453,360	B1 *	9/2002	Muller et al.	709/250
6,542,991	B1 *	4/2003	Joy et al.	712/228
6,560,511	B1 *	5/2003	Yokoo et al.	700/245
6,571,278	B1 *	5/2003	Negishi et al.	709/213
6,606,645	B1 *	8/2003	Cohen et al.	709/203
6,647,534	B1	11/2003	Graham	
6,993,591	B1	1/2006	Klemm	
6,999,717	B2 *	2/2006	Spratt et al.	455/7
7,003,582	B2 *	2/2006	Basso et al.	709/242

(Continued)

FOREIGN PATENT DOCUMENTS

WO	WO 99/27680	6/1999

OTHER PUBLICATIONS

Cohen et al., "Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency", IEEE INFOCOM 2000, 2000, pp. 854-863.

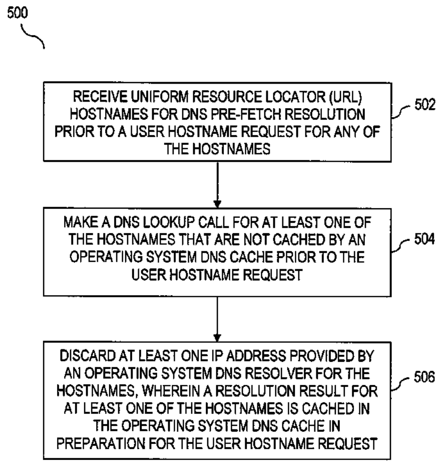
(Continued)

Primary Examiner — Dung B Huynh
(74) *Attorney, Agent, or Firm* — Sterne, Kessler, Goldstein & Fox P.L.L.C.

(57) ABSTRACT

Embodiments of the present invention include methods and systems for accelerated application startup. A method for accelerating startup of an application is provided. The method includes persistently storing a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application. The method further includes, upon startup of the application, making a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by the application, wherein a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request. A system for accelerating startup of an application is provided. The system includes a hostname storage device, a DNS pre-fetcher and a startup DNS pre-cacher.

23 Claims, 5 Drawing Sheets


500
RECEIVE UNIFORM RESOURCE LOCATOR (URL) HOSTNAMES FOR DNS PRE-FETCH RESOLUTION PRIOR TO A USER HOSTNAME REQUEST FOR ANY OF THE HOSTNAMES
502
MAKE A DNS LOOKUP CALL FOR AT LEAST ONE OF THE HOSTNAMES THAT ARE NOT CACHED BY AN OPERATING SYSTEM DNS CACHE PRIOR TO THE USER HOSTNAME REQUEST
504
DISCARD AT LEAST ONE IP ADDRESS PROVIDED BY AN OPERATING SYSTEM DNS RESOLVER FOR THE HOSTNAMES, WHEREIN A RESOLUTION RESULT FOR AT LEAST ONE OF THE HOSTNAMES IS CACHED IN THE OPERATING SYSTEM DNS CACHE IN PREPARATION FOR THE USER HOSTNAME REQUEST
506

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,343,397	B2 *	3/2008	Kochanski	709/218
7,356,534	B2	4/2008	Mohammed et al.	
7,970,891	B1 *	6/2011	Kontothanssis et al.	709/224
8,527,492	B1 *	9/2013	Issa	707/707
2001/0023459	A1 *	9/2001	Asami	709/245
2001/0043600	A1 *	11/2001	Chatterjee et al.	370/390
2001/0054045	A1 *	12/2001	Shirasaka	707/204
2002/0133487	A1 *	9/2002	Oshins et al.	707/5
2003/0028591	A1 *	2/2003	Goloshubin et al.	709/203
2003/0037254	A1 *	2/2003	Fischer et al.	713/200
2003/0067923	A1 *	4/2003	Ju et al.	370/395.3
2003/0093461	A1	5/2003	Suzuki et al.	
2003/0212732	A1 *	11/2003	Edahiro et al.	709/106
2003/0229673	A1 *	12/2003	Malik	709/207
2003/0236771	A1 *	12/2003	Becker	707/2
2004/0107278	A1	6/2004	Emaru et al.	
2005/0018249	A1 *	1/2005	Miura et al.	358/1.15
2005/0027892	A1 *	2/2005	McCabe et al.	709/253
2005/0086194	A1 *	4/2005	Suzuki et al.	707/1
2005/0114485	A1	5/2005	McCollum	
2005/0120180	A1	6/2005	Schornbach et al.	
2005/0235044	A1 *	10/2005	Tazuma	709/217
2005/0262248	A1 *	11/2005	Jennings et al.	709/228
2005/0286510	A1 *	12/2005	Nakajima et al.	370/386
2006/0069746	A1 *	3/2006	Davis et al.	709/218
2006/0129677	A1 *	6/2006	Tamura	709/227
2006/0242242	A1 *	10/2006	Ezumi et al.	709/206
2006/0248195	A1 *	11/2006	Toumura et al.	709/226
2006/0271642	A1 *	11/2006	Stavrakos et al.	709/217
2006/0294223	A1 *	12/2006	Glasgow et al.	709/224
2007/0050491	A1 *	3/2007	Kataoka et al.	709/223
2007/0061465	A1 *	3/2007	Kim et al.	709/226
2007/0294419	A1	12/2007	Ulevitch	
2008/0177894	A1 *	7/2008	Jennings et al.	709/232
2009/0103126	A1 *	4/2009	Park	358/1.15
2009/0171930	A1	7/2009	Vaughan et al.	
2009/0198779	A1	8/2009	Agrawal et al.	
2009/0222584	A1 *	9/2009	Josefsberg et al.	709/245
2009/0292696	A1 *	11/2009	Shuster	707/5
2010/0011053	A1 *	1/2010	Bhogal et al.	709/203
2010/0049872	A1	2/2010	Roskind	
2010/0114822	A1	5/2010	Pollock et al.	
2010/0146415	A1	6/2010	Lepeska	
2010/0154055	A1	6/2010	Hansen	
2012/0084852	A1 *	4/2012	Ong	726/12

OTHER PUBLICATIONS

Klemm, "WebCompanion: A Friendly Client-Side Web Prefetching Agent", IEEE Transactions on Knowledge and Data Engineering vol. 11, No. 4, Jul./Aug. 1999, pp. 577-594.

Wang, et al., "Prefetching in World Wide Web", IEEE, 1996, pp. 28-32.

Bouras, C., et al., "Predictive Prefetching on the Web and its Potential Impact in the Wide Area," *World Wide Web: Internet and Web Information Systems*, 2004, vol. 7, pp. 143-179; Kluwer Academic Publishers, The Netherlands.

PCT, Notification of Transmittal of the International Search Report and the Written Opinion of the International Searching Authority, or the Declaration, along with the PCT Written Opinion of the International Searching Authority for International Appln. No. PCT/US2009/055375; International Filing Date: Aug. 28, 2009; 12 pages.

EP Patent Office, PCT—Notification of Transmittal of the International Search Report and the Written Opinion of the International Searching Authority, or the Declaration, PCT—International Search Report, and PCT—Written Opinion of the International Search Authority; International Appln. No. PCT/US2009/054906; International Filing Date: Aug. 25, 2009, 13 pages.

Office Communication, dated Jan. 19, 2010, for U.S. Appl. No. 12/197,907, filed Aug. 25, 2008, 19 pages.

Office Communication, dated Jul. 8, 2010, for U.S. Appl. No. 12/197,907, filed Aug. 25, 2008, 19 pages.

Office Communication, dated Dec. 21, 2010, for U.S. Appl. No. 12/197,907, filed Aug. 25, 2008, 32 pages.

Office Communication, dated Apr. 12, 2011, for U.S. Appl. No. 12/197,907, filed Aug. 25, 2008, 3 pages.

Office Communication, dated Aug. 25, 2011, for U.S. Appl. No. 12/197,907, filed Aug. 25, 2008, 24 pages.

Office Communication, dated Dec. 14, 2010, for U.S. Appl. No 12/415,471, filed Mar. 31, 2009, 11 pages.

Office Communication, dated Aug. 2, 2011, for U.S. Appl. No. 12/415,471, filed Mar. 31, 2009, 17 pages.

U.S. Appl. No. 12/415,471, filed Mar. 31, 2009, 24 pages.

* cited by examiner

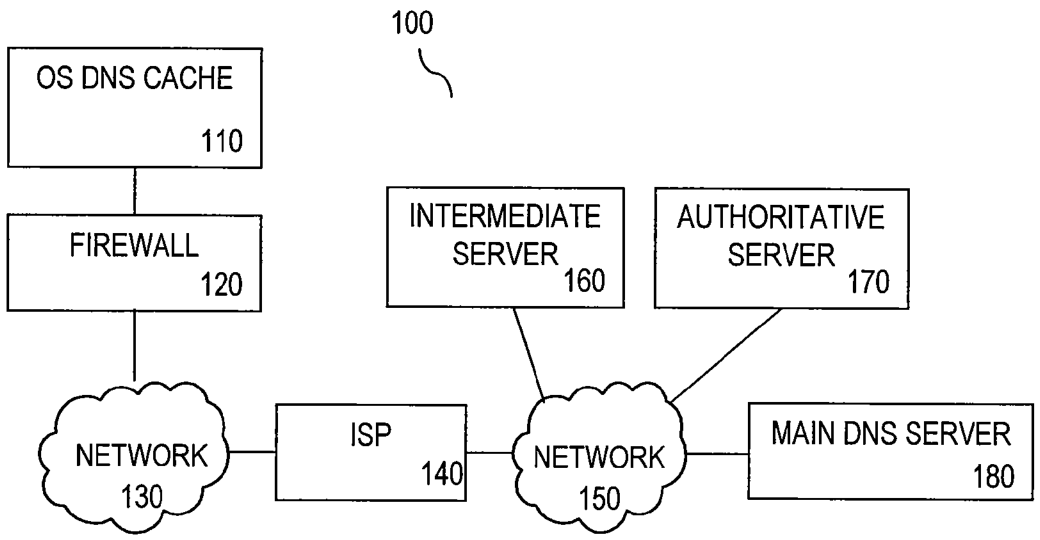

100
OS DNS CACHE 110
FIREWALL 120
INTERMEDIATE SERVER 160
AUTHORITATIVE SERVER 170
NETWORK 130
ISP 140
NETWORK 150
MAIN DNS SERVER 180

FIG. 1

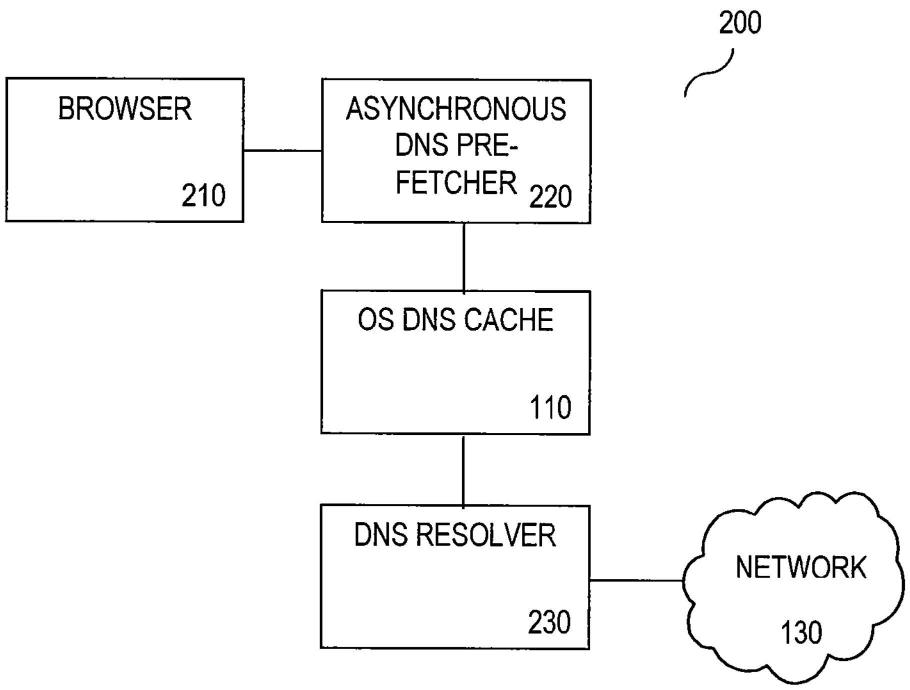

200
BROWSER 210
ASYNCHRONOUS DNS PRE-FETCHER 220
OS DNS CACHE 110
DNS RESOLVER 230
NETWORK 130

FIG. 2

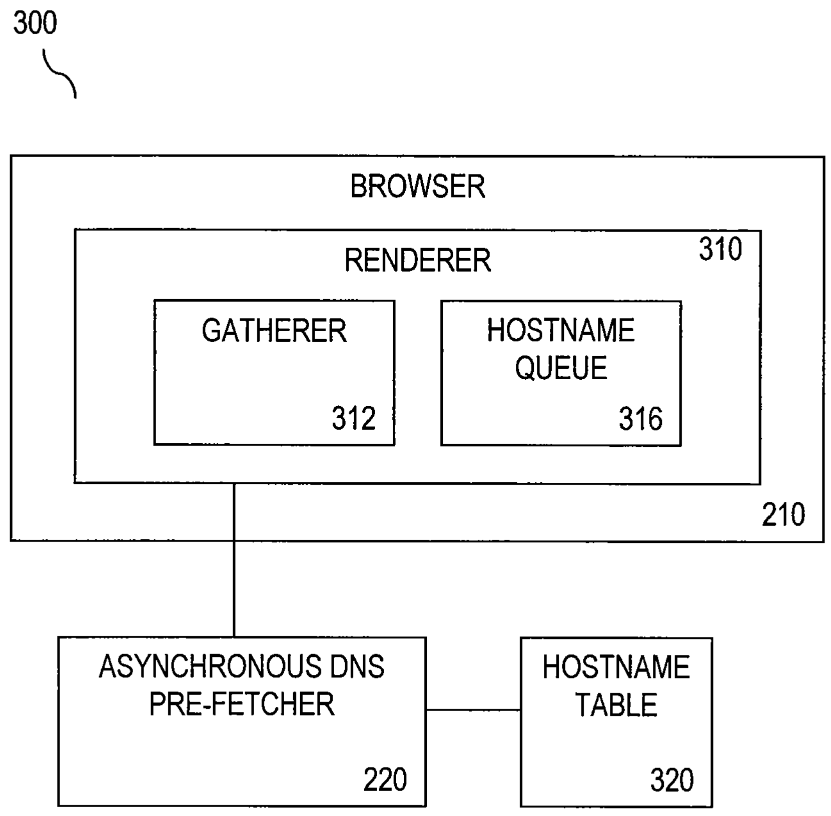

300
BROWSER
RENDERER
310
GATHERER
312
HOSTNAME QUEUE
316
210
ASYNCHRONOUS DNS PRE-FETCHER
220
HOSTNAME TABLE
320

FIG. 3

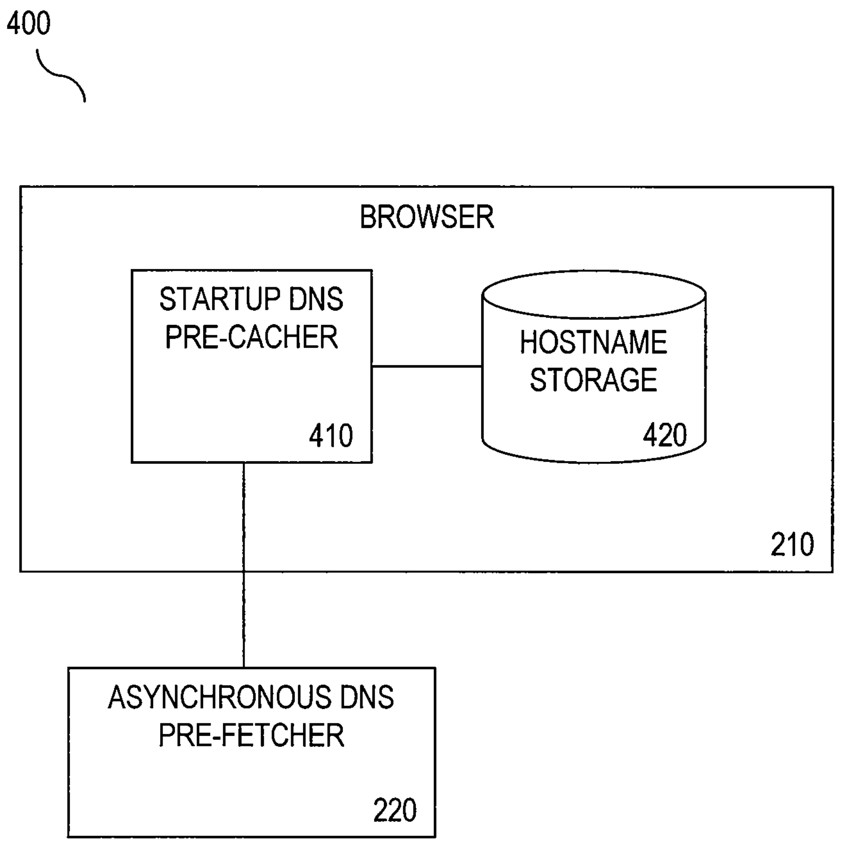

400
BROWSER
STARTUP DNS PRE-CACHER
410
HOSTNAME STORAGE
420
210
ASYNCHRONOUS DNS PRE-FETCHER
220

FIG. 4

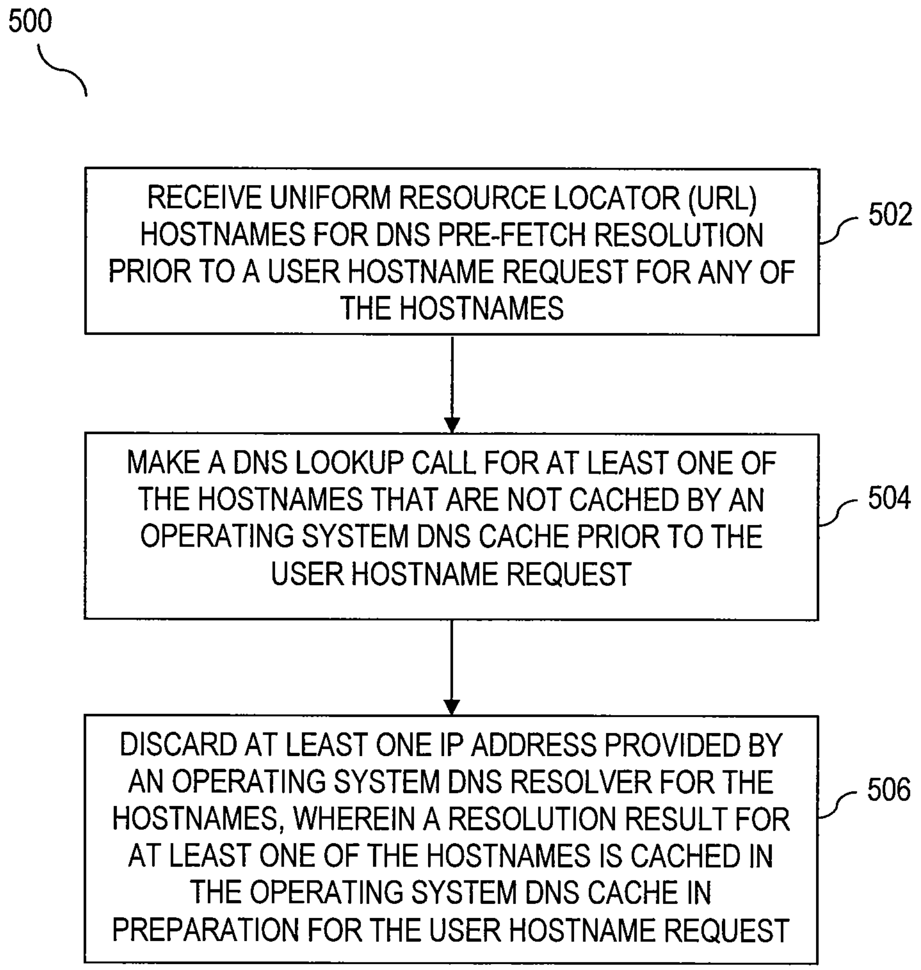

500
RECEIVE UNIFORM RESOURCE LOCATOR (URL) HOSTNAMES FOR DNS PRE-FETCH RESOLUTION PRIOR TO A USER HOSTNAME REQUEST FOR ANY OF THE HOSTNAMES
502
MAKE A DNS LOOKUP CALL FOR AT LEAST ONE OF THE HOSTNAMES THAT ARE NOT CACHED BY AN OPERATING SYSTEM DNS CACHE PRIOR TO THE USER HOSTNAME REQUEST
504
DISCARD AT LEAST ONE IP ADDRESS PROVIDED BY AN OPERATING SYSTEM DNS RESOLVER FOR THE HOSTNAMES, WHEREIN A RESOLUTION RESULT FOR AT LEAST ONE OF THE HOSTNAMES IS CACHED IN THE OPERATING SYSTEM DNS CACHE IN PREPARATION FOR THE USER HOSTNAME REQUEST
506

FIG. 5

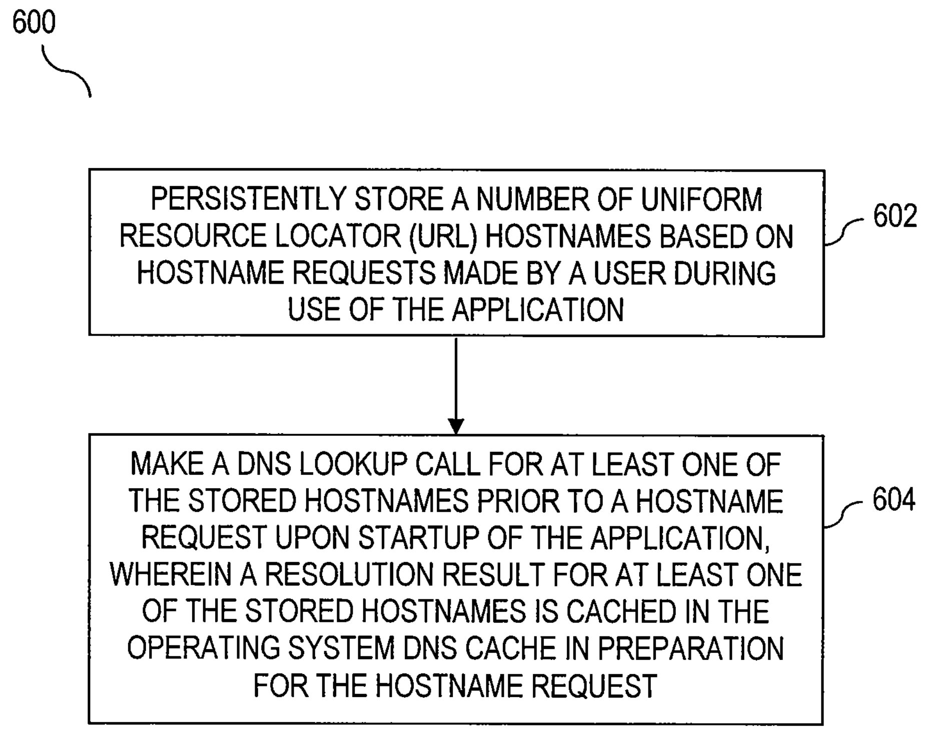

600
PERSISTENTLY STORE A NUMBER OF UNIFORM RESOURCE LOCATOR (URL) HOSTNAMES BASED ON HOSTNAME REQUESTS MADE BY A USER DURING USE OF THE APPLICATION
602
MAKE A DNS LOOKUP CALL FOR AT LEAST ONE OF THE STORED HOSTNAMES PRIOR TO A HOSTNAME REQUEST UPON STARTUP OF THE APPLICATION, WHEREIN A RESOLUTION RESULT FOR AT LEAST ONE OF THE STORED HOSTNAMES IS CACHED IN THE OPERATING SYSTEM DNS CACHE IN PREPARATION FOR THE HOSTNAME REQUEST
604

FIG. 6

ADAPTIVE ACCELERATED APPLICATION STARTUP

BACKGROUND

1. Field of the Invention

Embodiments of the present invention relate to applications and the World Wide Web.

2. Background Art

A web browser is a software application that allows a user to view or download content that is available on a network, such as on a website on the World Wide Web. Content may include text, files, images, audio, video and personal communications. A browser may also allow a user to enter, upload, or execute content. Browsers run on personal computers and mobile devices. Commonly used browsers may presently include, for example, FIREFOX, INTERNET EXPLORER, SAFARI, and OPERA.

Browsers may use a number of protocols and standards to obtain or manage content flow. Most browsers primarily use hypertext transfer protocol (HTTP) to fetch content and webpages. Webpages are located using a uniform resource locator (URL), which identifies where the webpage may be found. Webpages may be retrieved using the IP address of the computer holding the webpage content. In order to be more memorable and human friendly, an IP address or hierarchy may be represented by a hostname (such as www.google.com). A hostname is a domain name that has one or more associated IP addresses. A hostname request is a request by a user to navigate to a webpage using a URL hostname. For example, a hostname request may include a user clicking on a link on a web page or typing a hostname in a URL bar. Hostnames and other information associated with domain names may be resolved or translated to IP addresses using the Domain Name System (DNS). This DNS resolution system is sometimes referred to as the “phone book” for the Internet.

DNS resolution requires either looking in a local computer cache or querying a set of DNS servers over the network. A request for DNS resolution may also be known as a DNS lookup call. DNS utilizes authoritative name servers to help map domain names to IP addresses in order to avoid having all the information in a single, central DNS server. These and other intermediate name servers may cache DNS resolution information to shorten DNS resolution times.

For example, FIG. **1** illustrates an exemplary system **100** that performs DNS resolution. When network traffic is required, UDP packets are sent to a DNS resolver, and eventually a UDP response is provided. DNS resolutions may exist in a local cache, such as operating system DNS cache **110**. If not, the next resolver is commonly LAN firewall **120**, which necessitates traffic from the firewall resolver to another resolver, such as ISP **140**, over network **130**. The latency time of two such round trips may presently be no less than 40 ms compared to 0-3 ms when operating system DNS cache **110** is the source of the resolution. If resolution information is not in the cache of firewall **120** or ISP **140**, other intermediate servers **160** may be queried over one or more networks **150**. If the hostname is yet to be resolved, authoritative server **170** or main DNS server **180** will be queried and latency will be further increased. Failures, delays and lost packets contribute to accumulated latency that can commonly exceed 1 second or longer. Longer latency times cause discomfort to users of a web browser.

DNS resolution times can be reduced. When DNS resolution occurs for a website, cached results will make future visits to a website quicker. For instance, a web page when first visited may have a portion of its presentation latency attributable to DNS resolution, which could exceed 120 milliseconds. Future visits will get DNS queries from cache at no cost.

User-perceived latency may be reduced through DNS pre-fetching. DNS pre-fetching resolves or fetches a variety of hostnames through the DNS in advance of user activities, anticipating that one of those name resolutions will probably be useful in an upcoming user webpage or hostname request. However, browsers currently do not do DNS pre-fetching for a number of reasons. Engineers have not implemented techniques for DNS pre-fetching in browsers, fearing that the delicate complexity of the network stack would be compromised. Also, engineers have thought that implementations would have to be adapted for each different network application or browser. Further, engineers have worried that any additional network code, processing or complexity prior to a user request would only further increase latency.

BRIEF SUMMARY

Embodiments described herein refer to systems and methods for domain name system (DNS) pre-caching. Embodiments described herein also refer systems and methods for DNS pre-caching for accelerating application startup. According to an embodiment, a method for accelerating startup of an application is provided. The method includes persistently storing a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application. The method also includes, upon startup of the application, making a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by the application, wherein a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request.

According to another embodiment, a method for accelerating startup of an application is provided. The method includes receiving a number of stored uniform resource locator (URL) hostnames. The method also includes, upon startup of the application, making a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by the application. The method further includes discarding at least one IP address provided by an operating system DNS resolver for the stored hostnames, wherein a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request.

According to a further embodiment, a system for accelerated application startup is provided. The system includes a hostname storage device configured to persistently store a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application. The system also includes a DNS pre-fetcher configured to make a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by an application for any of the stored hostnames, wherein a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request. The system further includes a startup DNS pre-cacher configured to pass the number of hostnames to the DNS pre-fetcher from the hostname storage device upon startup of the application.

Further embodiments, features, and advantages of the invention, as well as the structure and operation of the various embodiments of the invention are described in detail below with reference to accompanying drawings.

BRIEF DESCRIPTION OF THE FIGURES

Embodiments of the invention are described with reference to the accompanying drawings. In the drawings, like reference numbers may indicate identical or functionally similar elements. The drawing in which an element first appears is generally indicated by the left-most digit in the corresponding reference number.

FIG. **1** is a diagram showing an existing system for DNS hostname resolution, according to an embodiment of the present invention.

FIG. **2** is a diagram of a system for DNS pre-caching, according to an embodiment of the present invention.

FIG. **3** is a more detailed diagram of a system for DNS pre-caching, according to an embodiment of the present invention.

FIG. **4** is a diagram of a system for accelerating application startup using DNS pre-caching, according to an embodiment of the present invention.

FIG. **5** is a flowchart illustrating a method for DNS pre-caching, according to an embodiment of the present invention.

FIG. **6** is a flowchart illustrating a method for accelerating application startup using DNS pre-caching, according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

While the present invention is described herein with reference to illustrative embodiments for particular applications, it should be understood that the invention is not limited thereto. Those skilled in the art with access to the teachings provided herein will recognize additional modifications, applications, and embodiments within the scope thereof and additional fields in which the invention would be of significant utility.

The goal of Domain Name System (DNS) pre-fetching is to reduce user-perceived latency when surfing the Internet. Perceived network code complexity has prevented an effective implementation of DNS pre-fetching in browsers. A simpler implementation does not involve creating an additional pre-fetching DNS cache, but rather “pre-warming up” an existing DNS cache for actual browser network stack accesses. A cache is considered “warm” if it happens to contain a subset of data that later proves useful. Usually a cache is warmed up by actual use in a process and not by a system deliberately trying to pre-populate a cache. Loading a cache is often expensive and cache size may be restricted, causing some data to be evicted in favor of other data without proof of utility of the new data. With DNS caching, according to many embodiments of the present invention, cache size is not a significant restriction.

One or more parallel threads may be executed to perform hostname-to-IP address DNS resolution. Such a thread can process a hostname, as though it were looking the IP address up, but not actually obtain and use the IP address to render the web page for the IP address. In fact, results of such lookups may be discarded, or deleted, in some implementations. Instead, the act of calling a DNS query function, which is often passed to the operating system, causes IP resolutions to be stored in a DNS lookup system’s cache. This is “pre-caching”. Later, before a DNS cache expires, a thread can fetch the resource (e.g., URL) and the underlying cache will cause that fetch to be expedited. As a result, there will be little or no latency delay in obtaining a DNS resolution since it is already in the cache. This leads to dramatic savings in later URL navigation. Embodiments of the present invention include methods and systems for domain name system (DNS) pre-caching.

According to a feature, DNS resolutions are pre-cached as a “side effect” of parallel DNS queries. As described above, parallel threads may be executed to do DNS resolution as though they were looking the IP addresses up. This action, as a “side-effect”, loads a process and local cache (e.g. operating system cache) with DNS resolutions. Because such an implementation benefits from the “side-effect” of a normal DNS resolution query, it avoids the complexity of interacting with the traditional network stack that is resolving names. Latency measurements have produced significant and surprising positive results.

Hostnames for DNS pre-caching may be selected or gathered in various ways, increasing the utility of DNS pre-caching. Such DNS pre-caching can make better use of extra CPU time that exists while a browser is waiting for a user to select a link. DNS pre-caching may also be used to accelerate application startup. Such utilizations of DNS pre-caching will be discussed in further embodiments herein.

FIG. **2** illustrates an exemplary system **200** for DNS pre-caching, according to an embodiment. System **200** includes browser **210**, asynchronous DNS pre-fetcher **220**, operating system DNS cache **110** and DNS resolver **230**. These components may be coupled directly or indirectly, such as over a network. DNS resolver **230** may be coupled to one or more networks **130**. According to an embodiment, browser **210** may include any device, application or module that enables a user or computer to navigate and/or retrieve data from another data source, typically over a network. Browser **210** may include any conventional browser including but not limited to for example, FIREFOX available from Mozilla Foundation Inc., INTERNET EXPLORER available from Microsoft Corp., SAFARI available from Apple Computer, Inc., and OPERA available from Opera Software ASA. Browser **210** may also be a multi-process browser as described in “Multi-Process Browser Architecture,” by Fisher et al., U.S. Provisional Appl. No. 61/052,719, filed May 13, 2008. According to a further embodiment, browser **210** may also be configured to use any number of protocols, including protocols such as HTTP.

Browser **210** may exist within or be executed by hardware in a computing device. For example, browser **210** may be software, firmware, or hardware or any combination thereof in a computing device. A computing device can be any type of computing device having one or more processors. For example, a computing device can be a workstation, mobile device (e.g., a mobile phone, personal digital assistant, or laptop), computer, game console, set-top box, kiosk, embedded system or other device having at least one processor and memory.

According to an embodiment, browser **210** may be configured to receive one or more uniform resource locator (URL) hostnames for DNS pre-fetch resolution prior to a user hostname request for any of the one or more URL hostnames. A user hostname request may include, but is not limited to, clicking on a link, or pressing “Enter” or “Return” upon entering or selecting a URL address. A user hostname request may be any action or gesture by a user confirming or committing to a URL address or hostname. Browser **210** may also be configured to make a DNS lookup call for at least one of the one or more URL hostnames that are not cached by a DNS cache prior to the user hostname request. Browser **210** may be further configured to discard at least one IP address provided by a DNS resolver for the one or more URL hostnames, where a resolution result for at least one of the one or more URL hostnames is cached in a DNS cache in preparation for the user hostname request. According to a further embodiment, browser **210** may provide one or more URL hostnames to asynchronous DNS pre-fetcher.

According to an embodiment, asynchronous DNS pre-fetcher **220** may be configured to make a DNS lookup call for one or more uniform resource locator (URL) hostnames and discard an IP address for at least one of the URL hostnames prior to a user hostname request for any of the URL hostnames, where a resolution result for at least one of the URL hostnames is cached in a DNS cache in preparation for the user hostname request. Many operating systems provide DNS resolution service asynchronously. That is, rather than calling a function and "waiting" until it returns, a function may be called and it will "call back" when it has an answer. As a result, it can be called with many requests (while prior requests are still pending).

Asynchronous DNS pre-fetcher **220** may act in a similar asynchronous fashion. Also, DNS pre-fetcher **220** may act as a module separate from the network stack for DNS resolution. This may lead to a simpler implementation.

If DNS resolutions are not cached in OS DNS cache **110**, hostnames may be provided to DNS resolver **230**. DNS resolver **230** may require further queries over one or more networks **130** to resolve the hostnames provided by DNS pre-fetcher **220**.

FIG. **3** illustrates an exemplary system **300** for DNS pre-caching, according to an embodiment. System **300** includes browser **210**, asynchronous DNS pre-fetcher **220** and hostname table. System **300** may also include renderer **310**, which may or may not exist in browser **210**. These components may be coupled together directly or indirectly.

According to an embodiment, hostname table **320** may be a data structure configured to store DNS pre-fetch information for one or more URL hostnames. DNS pre-fetch information may include, but is not limited to: DNS resolution tracking information (is DNS task queued, assigned, resolved, etc.?) for each hostname; transition time for such tasks; when was the hostname resolved last; how many other resolutions have taken place since a hostname's last resolution (useful for estimating cache eviction); and a central hash table for hostnames and DNS resolution events. DNS pre-fetch information may also include information for services relating to: startup and teardown, enabling and disabling DNS pre-fetching, providing global entry points (requiring no context or instance) for DNS resolution requests, and monitoring and measuring performance of actual network stack resolutions required for web navigation. Any combination of the above DNS pre-fetch information may be included in other components of system **300**. According to a further embodiment, a cache eviction time may be determined by DNS pre-fetcher **220** based on DNS pre-fetch information in hostname table **320**. DNS pre-fetcher **220** may be further configured to make DNS lookup calls for only the one or more URL hostnames that have not had a DNS lookup call within a determined cache eviction time.

Renderer **310** may be a module that displays (or renders) data, such as an HTML page, according to an embodiment. Renderer **310**, in the course of analyzing a page, may ask the environment about its context. One example of context is whether a link was already visited. Such context may be used to identify possible links to gather for pre-fetching. According to an embodiment, renderer **310** may be configured to gather hostnames.

Hostnames gathered or captured by renderer **310** may be placed, either temporarily or persistently, into hostname queue **316** by renderer **310** or browser **210**, according to an embodiment. Hostname queue **316** may be a data structure or an allocation in memory. Hostname queue **316** may exist in or be provided by renderer **310** or browser **210**. Hostname queue **316** may be dynamic or static. According to an embodiment, when an individual or dedicated worker thread is available, it may gather one or more hostnames from hostname queue **316** for processing. A browser or renderer worker thread may make a blocking DNS lookup call for an assigned hostname and wait until a resolution is returned. Once there is a resolution (or name-not-found result), the worker thread may update hostname information in hostname table **320**. Such information can prevent pre-fetching the same name too often. If an IP address is provided by DNS resolver **230**, it may be discarded, according to a further embodiment. Having a collection of worker threads will prevent a slow DNS resolution from holding up quicker asynchronous resolutions from the remainder of hostname queue **316**. Browser **210**, DNS pre-fetcher **220**, or renderer **310** may perform tasks described above with one or more parallel threads.

Hostnames for DNS pre-fetching may be selected or determined in a number of ways. According to an embodiment, browser **210** or renderer **310** may include gatherer **312**. Gatherer **312** may be configured to gather one or more URL hostnames from one or more URL links in a web page prior to a user selecting a URL link in the web page. A web page may be scanned for links on the page. In some cases, all links on a page may be gathered. Limitations can be placed on how many hostnames are gathered from a webpage in cases where there are a large number of links. In these cases, renderer **310** may be configured to limit URL hostnames that are passed based upon a number of URL links on a webpage. It could be counter-productive if valuable hostnames where evicted from the DNS cache by hostname links that are less likely to be selected. In other cases, gatherer **312** may be configured to avoid duplicate hostnames. According to an embodiment, anchor tags (links that are colored to indicate whether the link was visited or not) may be parsed and hostnames extracted. According to a further embodiment, gatherer **312** may gather hostnames as seen on a "results" page from a search.

According to an embodiment, gatherer **312** may be configured to gather one or more URL hostnames from one or more predicted hostnames based upon entering activity in a URL address bar prior to a user completely entering a hostname. Entering activity may include a user typing a URL address in the address bar. Entering activity may also include plausible hostnames that are predicted or proposed to a user based on autocompletion logic, while the user is typing. Autocompletion proposals may be based on previously entered URLs, query submissions, or immediate termination of typing by the user (completion of explicit user URL entry). Each proposed hostname may be processed and resolved before a user finalizes his or her entry, whether or not a user ultimately agrees with a self-completing prediction.

Some links and advertisements may not contain a true URL link but only the URL of a server redirector. According to an embodiment, a link tag on a page may be augmented with a hint to the browser that a given domain will be visited. Observed link tags may cause a corresponding hostname to be inserted into hostname queue **316**. According to a further embodiment, a relationship value in a link tag associated with at least one of the one or more URL hostnames may be replaced. A replacement value may be associated with DNS pre-fetching. For example, a link tag may appear as "<link rel="DNS-pre-fetch" href=http://www.hostnametoprefeth.com/> Such actions may be performed by renderer **310**, gatherer **312**, or browser **210**.

According to an embodiment, gatherer **312** may be configured to gather one or more URL hostnames based upon one or more omni-box suggestions that appear based on user activity prior to a user selecting an omni-box suggestion. An omni-box is a query box provided by browser **210** that assists with routine surfing. An omni-box may suggest to a user (via a highlighted line) that a certain course of action be taken. For example, actions may include visiting a link that was visited in the past, doing a search provider query, or visiting a new URL. When such a suggestion is made, a potential URL is constructed. That URL hostname may be gathered by gatherer **312**. According to a further embodiment, a user may enter a search query in an onmi-box. A hostname may be fully resolved between when a user enters a URL and when he or she presses "Enter". When browser **210** determines a search will be performed, it forms a search URL and a search provider's hostname is gathered by gatherer **312**. In some cases, search time may be reduced by 120 ms.

According to an embodiment, gatherer **312** may be configured to gather one or more URL hostnames based upon one or more URL links on a web page that a user has moused over but has not selected. In some cases, this may be based on a period of time in which a user has a mouse pointer resting on a link or in proximity to a link. The hostname for the link may be pre-fetched while a user considers clicking on the link.

According to a further embodiment, renderer **310** may be configured to pass one or more URL hostnames to DNS pre-fetcher **220** or browser **210**. Hostnames may also be passed in an array. In another embodiment, each individual array transmission can be guaranteed to be internally duplicate free. Renderer **310** may be configured to avoid providing duplicate hostnames. Renderer **310** may also be configured to avoid providing variations on hostnames that would lead DNS pre-fetcher **220** to make superfluous DNS lookup calls. According to another embodiment, an API may pre-populate a DNS cache.

DNS pre-caching may be used for other purposes such as reducing application startup time. FIG. **4** illustrates an exemplary system **400** for accelerating application startup, according to an embodiment. System **400** includes browser **210**, startup DNS pre-cacher **410**, asynchronous DNS pre-fetcher **220** and hostname storage **420**. These components may be coupled together directly or indirectly. Startup DNS pre-cacher **410** and hostname storage **420** may exist in browser **210**.

According to an embodiment, hostname storage **420** may be a storage device configured to persistently store a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of an application. Hostname storage **420** may store hostnames requested by one or more applications. The selection of hostnames to be stored may be an adaptive process. Startup DNS pre-cacher **410** may monitor all URL fetches that are made during startup that involve network activity. According to a further embodiment, only a select subset of URL fetches may be monitored. For example, monitoring may exclude resolutions that are already cached and require no network activity. Monitoring may also exclude resolutions of names noted in a pre-specified list of names to ignore. Startup DNS pre-cacher may or may not be a component of browser **210**. The number of stored hostnames may or may not be predetermined.

According to an embodiment, DNS pre-cacher **410** may store a number of hostnames based upon a number of first hostnames requested by one or more users following startup of the application. For example, the first ten hostnames found in URLs may be stored in hostname storage **420**. According to a further embodiment, a variable number of hostnames may be stored in hostname storage **420**. For example, all hostnames resolved in the first N seconds, such as the first 10 seconds, may be stored. As another example, all hostnames requiring resolution in the first 5 minutes, with resolution time greater than some significant delay, such as 500 ms, may be stored. In some embodiments, a variable number of hostnames may be bounded by a fixed limit, for example, not to exceed 10 names.

According to another example, a number of hostnames based upon a number of most recent hostnames requested by one or more users during use of the application may be stored in hostname storage **420**. In a further example, a number of hostnames based upon a number of most frequently requested hostnames by one or more users during use of the application may be stored. In another example, a number of hostnames based upon any combination of first hostnames following startup, most recent hostnames, or most frequent hostnames requested by one or more users during use of the application may be stored. According to another embodiment, some hostnames stored in hostname storage **420** may be preprogrammed hostnames. Stored hostnames may also be generated or received from another application.

According to an embodiment, startup DNS pre-cacher **410** may be configured to pass a number of hostnames to asynchronous DNS pre-fetcher **220** from the hostname storage device upon startup of the application. For example, such a start up may be when the application is opened or selected to run. This start up may be the first time the application is started. This start may also be a startup subsequent to a closing of the application (also called a next start up). In other words, an earlier run of the application may have occurred and the application may have been previously closed. In this case, a start up may be a next start up subsequent to a closing of the application. DNS pre-fetcher **220** may be configured to make a DNS lookup call for at least one stored hostname prior to a hostname request for any of the stored hostnames, wherein a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request. According to an embodiment, hostname requests may be initiated by an application. An application may perform DNS lookups during startup in anticipation of hostname requests by the application. Such hostname requests may take place towards the end of the startup or at some time following startup. DNS lookups may also be made by the application in advance of any predicted user hostname requests. According to another embodiment, hostname requests may initiated by the application in response to a user gesture or a user hostname request. According to a further embodiment, DNS pre-fetcher **220** may be configured to discard at least one IP address provided by an operating system DNS resolver for stored hostnames, where a resolution result for at least one of the stored hostnames is cached in the operating system DNS cache in preparation for the hostname request.

Embodiments described herein for accelerated application startup reduce real-world application startup time. Without such an optimization, a user would first wait for the application to start, and then wait for a page to be fetched (including DNS lookup). Resolving hostnames in advance improves the application experience for the user. Hostnames that can be pre-fetched may also include a user's home page, domains commonly used in that page, or hostnames that are used in number of standard tabs that are typically loaded at startup. DNS pre-cacher **410** or DNS pre-fetcher **220** may be configured to perform tasks using one or more parallel threads.

FIG. **5** illustrates an exemplary method **500** for DNS pre-caching, according to an embodiment. In step **502**, URL hostnames for DNS pre-fetch resolution may be received prior to a user hostname request for any hostnames. In step **504**, a DNS lookup call for at least one of the URL hostnames that are not cached by a DNS cache is made prior to a user hostname request. In step **506**, at least one IP address provided by a DNS resolver for one or more URL hostnames may be discarded, where a resolution result for at least one of the URL hostnames is cached in a DNS cache in preparation for a user hostname request. According to an embodiment, steps **502**, **504** and **506** may be performed by DNS pre-fetcher **220**.

FIG. **6** illustrates an exemplary method **600** for accelerated application startup, according to an embodiment. In step **602**, a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application are persistently stored. According to an embodiment, DNS pre-cacher **410** may be configured to perform step **602**. Hostnames may be stored in hostname storage **420**. In step **604**, a DNS lookup call for at least one of the stored hostnames may be made prior to a hostname request initiated by the application, where a resolution result for at least one of the stored hostnames is cached in operating system DNS cache in preparation for the hostname request. This may be performed upon startup of the application. According to a further embodiment, this startup may be a next startup subsequent to a closing of the application. According to an embodiment, step **604** may be performed by DNS pre-fetcher **220**. According to another embodiment, step **604** may perform a DNS lookup call in preparation for a user hostname request.

Aspects of the present invention, for exemplary system **200**, system **300**, system **400**, method **500** and/or method **600** or any part(s) or function(s) thereof may be implemented using hardware, software modules, firmware, tangible computer readable media having instructions stored thereon, or a combination thereof and may be implemented in one or more computer systems or other processing systems.

The present invention has been described above with the aid of functional building blocks illustrating the implementation of specified functions and relationships thereof. The boundaries of these functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternate boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed.

The foregoing description of the specific embodiments will so fully reveal the general nature of the invention that others can, by applying knowledge within the skill of the art, readily modify and/or adapt for various applications such specific embodiments, without undue experimentation, without departing from the general concept of the present invention. Therefore, such adaptations and modifications are intended to be within the meaning and range of equivalents of the disclosed embodiments, based on the teaching and guidance presented herein. It is to be understood that the phraseology or terminology herein is for the purpose of description and not of limitation, such that the terminology or phraseology of the present specification is to be interpreted by the skilled artisan in light of the teachings and guidance.

The breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A method for accelerating startup of a web browser running on a computing device, comprising:

persistently storing, by the web browser running on the computing device, a number of uniform resource locator (URL) hostnames in a hostname storage of the web browser and based on hostname requests made by one or more users during a prior use of the web browser;

in response to an initial execution of an instance of the web browser and prior to a hostname request for any of the stored URL hostnames:

performing, by the web browser, hostname-to-IP address DNS resolutions for the stored URL hostnames;

receiving, by the web browser, resolution results for the stored URL hostnames in response to the performing step, the resolution results including IP addresses and hostname information for the stored URL hostnames;

caching, by the web browser, the hostname information for the received resolution results of the hostname-to-IP address DNS resolutions in an operating system DNS cache in preparation for the hostname request of the stored URL hostnames; and

discarding, by the web browser, the IP addresses of the received resolution results, such that subsequent hostname requests of the stored URL hostnames are processed (i) utilizing the hostname storage and operating system DNS cache and (ii) separate from network activity.

2. The method of claim **1**, wherein the performing, caching, and discarding steps are performed by parallel threads of the web browser.

3. The method of claim **1**, wherein the performing step further comprises:

inserting the stored URL hostnames into a work queue; and

performing a hostname-to-IP address DNS resolution for a next hostname in the work queue with an available work thread within a predetermined cache eviction time.

4. The method of claim **1**, wherein the storing step further comprises storing resolution information for the hostname in a hostname table having a hash table for hostnames and DNS resolution events.

5. The method of claim **1**, wherein the performing step includes making a DNS lookup call for a predicted hostname based on autocompletion logic or omnibox suggestions while a user is typing the hostname request and wherein the performing step is performed before the hostname request is finalized.

6. The method of claim **1**, wherein the storing step further comprises storing the number of URL hostnames based upon a number of first hostnames requested by the web browser following initial execution of the instance of the web browser.

7. The method of claim **1**, wherein the storing includes avoiding storing of duplicate URL hostnames.

8. The method of claim **1**, wherein the storing step further comprises storing the number of URL hostnames in a hostname queue based upon replaced values in link tags associated with the number of hostnames.

9. The method of claim **1**, wherein the storing step further comprises storing the number of URL hostnames based upon any combination of first hostnames following startup, most recent hostnames, or most frequent hostnames requested by one or more users during use of the web browser.

10. The method of claim **1**, further comprising discarding each IP address provided at startup by an operating system DNS resolver for the respective stored hostname.

11. A method for accelerating startup of a web browser running on a computing device, comprising:

receiving, by the web browser running on the computing device, a number of stored uniform resource locator (URL) hostnames;

in response to an initial execution of an instance of the web browser and prior to a hostname request for any of the stored URL hostnames:

performing, by the web browser, hostname-to-IP address DNS resolutions for the stored URL hostnames;

receiving, by the web browser, resolution results for the stored URL hostnames in response to the performing step, the resolution results including IP addresses and hostname information for the stored URL hostnames;

caching, by the web browser, the hostname information for the received resolution results of the hostname-to-IP address DNS resolutions in an operating system DNS cache in preparation for the hostname request of the received URL hostnames; and

discarding, by the web browser, the IP addresses received in response to the hostname-to-IP address DNS resolutions such that subsequent hostname requests of the stored URL hostnames are processed (i) utilizing the operating system DNS cache and (ii) separate from network activity.

12. The method of claim **11**, wherein the stored hostnames are preprogrammed hostnames.

13. The method of claim **11**, wherein the stored hostnames are generated by another application.

14. A system for accelerating startup of a web browser comprising:

a hostname storage device configured to persistently store a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during a prior use of the web browser;

a DNS pre-fetcher configured to

in response to an initial execution of an instance of the web browser and prior to a hostname request for any of the stored URL hostnames:

perform hostname-to-IP address DNS resolutions for the stored URL hostnames;

receive resolution results for the stored URL hostnames in response to the hostname-to-IP address DNS resolutions, the resolution results including IP addresses and hostname information for the stored URL hostnames;

cache the hostname information for the received resolution results of the hostname-to-IP address DNS resolutions in an operating system DNS cache in preparation for the hostname request of the stored URL hostnames; and

discard the IP addresses of the received resolution results, such that subsequent hostname requests of the stored URL hostnames are processed (i) utilizing the hostname storage and operating system DNS cache and (ii) separate from network activity; and

a startup DNS pre-cacher configured to pass the number of URL hostnames to the DNS pre-fetcher from the hostname storage device upon execution of the instance of the web browser.

15. The system of claim **14**, wherein the DNS pre-fetcher and the DNS pre-cacher are configured to operate using parallel threads of the web browser.

16. The system of claim **14**, further comprising a hostname table having a hash table for host names and corresponding DNS resolution events relating to providing global entry points for DNS resolution results, monitoring and measuring performance of actual network stack resolutions for web navigation, or DNS task assignments.

17. The system of claim **14**, wherein the DNS pre-cacher is further configured to store the number of URL hostnames based upon a number of first hostnames requested by one or more users following initial execution of the instance of the web browser.

18. The system of claim **14**, wherein the DNS pre-cacher is further configured to store the number of URL hostnames based upon a number of first hostnames requested by the web browser following initial execution of the instance of the web browser.

19. The system of claim **14**, wherein the DNS pre-cacher is further configured to store the number of URL hostnames based upon a number of most frequently requested hostnames by one or more users during use of the web browser.

20. The system of claim **14**, wherein the DNS pre-cacher is further configured to store the number of URL hostnames based upon any combination of first hostnames following initial execution of the instance of the web browser, most recent hostnames, or most frequent hostnames requested by one or more users during use of the web browser.

21. The system of claim **14**, wherein the DNS pre-fetcher is further configured to discard each IP address provided for a respective hostname at initial execution of the web browser, wherein each IP address is not retained by the web browser at initial execution.

22. A method for accelerating startup of an application running on a computing device, comprising:

persistently storing a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application;

upon startup of the application, performing a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by the application after startup, wherein a resolution result provided by a DNS resolver at startup for the at least one stored hostname is cached in an operating system DNS cache in preparation for the hostname request, the resolution result associated with IP addresses corresponding to the stored hostnames and wherein the performing further comprises: making a DNS lookup call for a predicted hostname based on autocompletion logic or omnibox suggestions while a user is typing the hostname request and wherein the performing step is performed before the hostname request is finalized; and

discarding the IP addresses corresponding to the hostnames while maintaining the cached resolution result provided by the DNS resolver at startup for the at least one stored hostname.

23. A system for accelerating startup of an application comprising:

a hostname storage device configured to persistently store a number of uniform resource locator (URL) hostnames based on one or more hostname requests made by one or more users during use of the application;

a DNS pre-fetcher configured to:

make a DNS lookup call for at least one of the stored hostnames prior to a hostname request initiated by the application for any of the stored hostnames and

discard IP addresses corresponding to the hostnames while maintaining a cached resolution result provided by a DNS resolver at startup for the at least one stored hostname, wherein the resolution result provided by the DNS resolver at startup for the at least one stored hostname is cached in the operating system DNS cache in preparation for the hostname request;

a startup DNS pre-cacher configured to pass the number of hostnames to the DNS pre-fetcher from the hostname storage device upon startup of the application; and

a memory comprising a hostname table having a hash table for host names and corresponding DNS resolution events relating to providing global entry points for DNS resolution results, monitoring and measuring performance of actual network stack resolutions for web navigation, or DNS task assignments.

* * * * *